

Teresa M. Corbin (SBN 132360)
 Christopher Kelley (SBN 166608)
 Thomas C. Mavrakakis (SBN 177927)
 HOWREY SIMON ARNOLD & WHITE, LLP
 301 Ravenswood Avenue
 Menlo Park, California 94025
 Telephone: (650) 463-8100
 Facsimile: (650) 463-8400
 Attorneys for Defendants
 MATROX ELECTRONIC SYSTEMS LTD. and
 MATROX GRAPHICS INC.

UNITED STATES DISTRICT COURT
 NORTHERN DISTRICT OF CALIFORNIA
 SAN FRANCISCO DIVISION

RICOH COMPANY, LTD.,

Plaintiff,

vs.

AEROFLEX INCORPORATED, AMI
 SEMICONDUCTOR, INC., MATROX
 ELECTRONIC SYSTEMS LTD., MATROX
 GRAPHICS INC., MATROX
 INTERNATIONAL CORP., and MATROX
 TECH, INC.,

Defendants.

) Case No.CV 03-04669 MJJ

)
) **DECLARATION OF ERIK K. MOLLER IN**
) **SUPPORT OF MATROX ELECTRONIC**
) **SYSTEMS, LTD. AND MATROX**
) **GRAPHICS INC.'S MOTION FOR**
) **SUMMARY JUDGMENT OF**
) **NONINFRINGEMENT**

) Date: December 23, 2003
) Time: 9:30 a.m.
) Ctrm: 11, 19th Floor
) Hon. Martin J. Jenkins
)

I, Erik K. Moller, declare as follows:

1. I am an attorney at law licensed to practice in the State of California and an associate of the law firm of Howrey Simon Arnold & White, LLP, attorneys for defendants Matrox Electronic Systems Ltd. and Matrox Graphics Inc. The matters set forth in this declaration are based upon my personal knowledge, except where otherwise indicated, and if called as a witness, I could and would testify competently thereto.

2. Attached hereto as Exhibit A is a true and correct copy of U.S. Patent No. 4,922,432.

3. Attached hereto as Exhibit B is a true and correct copy of the Memorandum and Order filed August 29, 2003 while this action was still pending in the United States District Court for the District of Delaware.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct. This declaration was executed in Menlo Park, California on December 2, 2003.

/s/

Erik K. Moller

- [54] **KNOWLEDGE BASED METHOD AND APPARATUS FOR DESIGNING INTEGRATED CIRCUITS USING FUNCTIONAL SPECIFICATIONS**
- [75] Inventors: Hideaki Kobayashi, Columbia, S.C.; Masahiro Shindo, Osaka, Japan
- [73] Assignees: International Chip Corporation, Columbia, S.C.; Ricoh Company, Ltd., Tokyo, Japan
- [21] Appl. No.: 143,821
- [22] Filed: Jan. 13, 1988
- [51] Int. Cl.⁵ G06F 15/60
- [52] U.S. Cl. 364/490; 364/489; 364/488; 364/521
- [58] Field of Search 364/488-491, 364/521, 300, 513

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,635,208	1/1987	Coleby et al.	364/491
4,638,442	1/1987	Bryant et al.	364/489
4,648,044	3/1987	Hardy et al.	364/513
4,651,284	3/1987	Watanabe et al.	364/491
4,656,603	4/1987	Dunn	364/488
4,658,370	4/1987	Erman et al.	364/513
4,675,829	6/1987	Clemenson	364/513
4,700,317	10/1987	Watanabe et al.	364/521
4,703,435	10/1987	Darringer et al.	364/488
4,803,636	2/1989	Nishiyama et al.	364/491

FOREIGN PATENT DOCUMENTS

1445914	8/1976	United Kingdom	364/490
---------	--------	----------------	---------

OTHER PUBLICATIONS

- "Verifying Compiled Silicon", by E. K. cheng, VLSI Design, Oct. 1984, pp. 1-4.
- "CAD System for IC Design", by M. E. Daniel et al., IEEE Trans. on Computer-Aided Design of Integrated Circuits & Systems, vol. CAD-1, No. 1, Jan. 1982, pp. 2-12.
- "An Overview of Logic Synthesis System", by L. Trevillyan, 24th ACM/IEEE Design Automation Conference, 1978, pp. 166-172.
- "Methods Used in an Automatic Logic Design Generator", by T. D. Friedman et al., IEEE Trans. on Computers, vol. C-18, No. 7, Jul. 1969, pp. 593-613.

- "Experiments in Logic Synthesis", by J. A. Darringer, IEEE ICC, 1980.
- "A Front End Graphic Interface to First Silicon Compiler", by J. H. Nash, EDA 84, Mar. 1984.
- "quality of Designs from An Automatic Logic Generator", by T. D. Friedman et al., IEEE 7th DA Conference, 1970, pp. 71-89.
- "A New Look at Logic Synthesis", by J. A. Darringer et al., IEEE 17th D. A. Conference 1980, pp. 543-548.
- Trevillyan—Trickey, H., *Flamel: A High Level Hardware Compiler*, IEEE Transactions On Computer Aided Design, Mar. 1987, pp. 259-269.
- Parker et al., *The CMU Design Automation System—An Example of Automated Data Path Design*, Proceedings Of The 16th Design Automation Conference, Las Vegas, Nev., 1979, pp. 73-80.
- An Engineering Approach to Digital Design*, William I. Fletcher, Prentice-Hall, Inc., pp. 491-505.

Primary Examiner—Felix D. Gruber

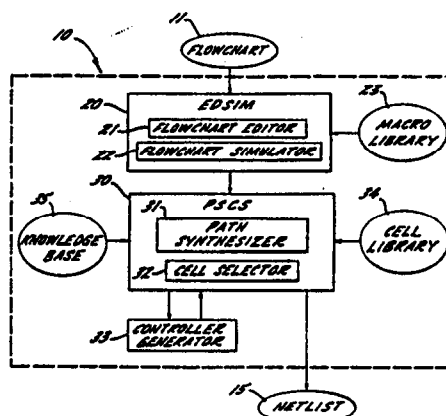
Assistant Examiner—V. N. Trans

Attorney, Agent, or Firm—Bell, Seltzer, Park & Gibson

[57] **ABSTRACT**

The present invention provides a computer-aided design system and method for designing an application specific integrated circuit which enables a user to define functional architecture independent specifications for the integrated circuit and which translates the functional architecture independent specifications into the detailed information needed for directly producing the integrated circuit. The functional architecture independent specifications of the desired integrated circuit can be defined at the functional architecture independent level in a flowchart format. From the flowchart, the system and method uses artificial intelligence and expert systems technology to generate a system controller, to select the necessary integrated circuit hardware cells needed to achieve the functional specifications, and to generate data and control paths for operation of the integrated circuit. This list of hardware cells and their interconnection requirements is set forth in a netlist. From the netlist it is possible using known manual techniques or existing VLSI CAD layout systems to generate the detailed chip level topological information (mask data) required to produce the particular application specific integrated circuit.

20 Claims, 12 Drawing Sheets



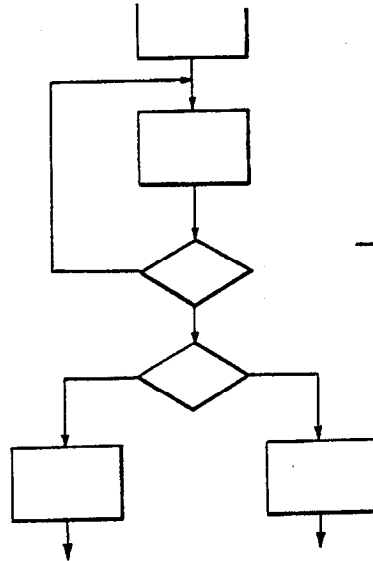


Fig. 1a.
FUNCTIONAL
LEVEL

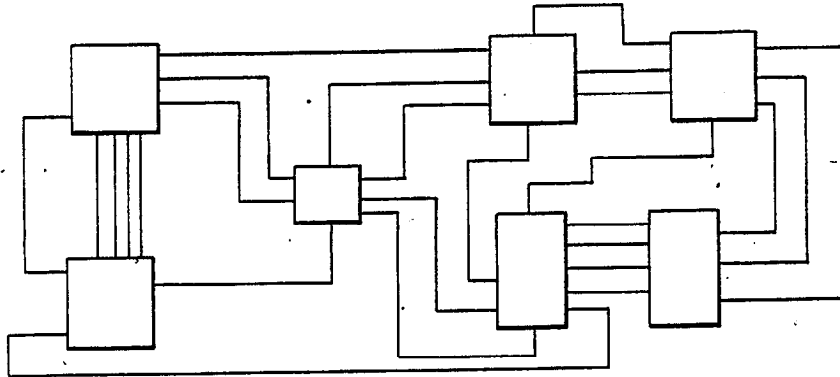


Fig. 1b.
STRUCTURAL LEVEL

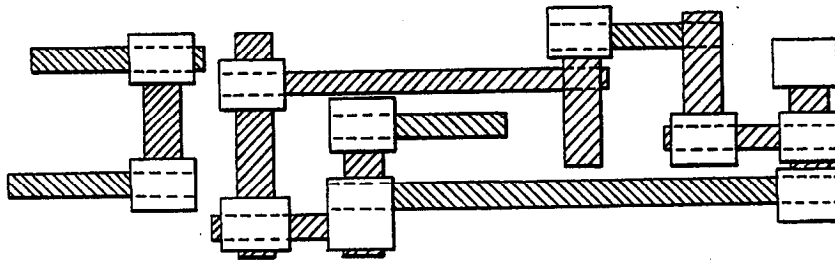


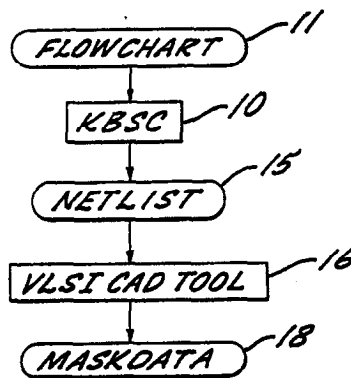
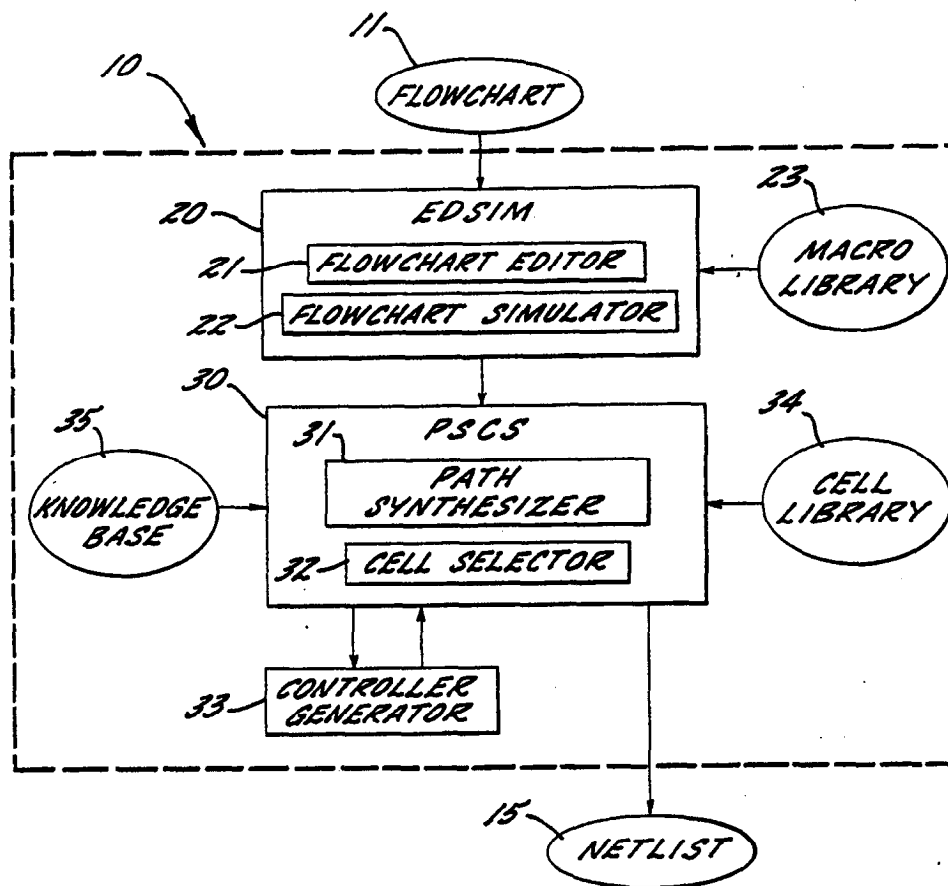
Fig. 1c.
PHYSICAL LAYOUT LEVEL

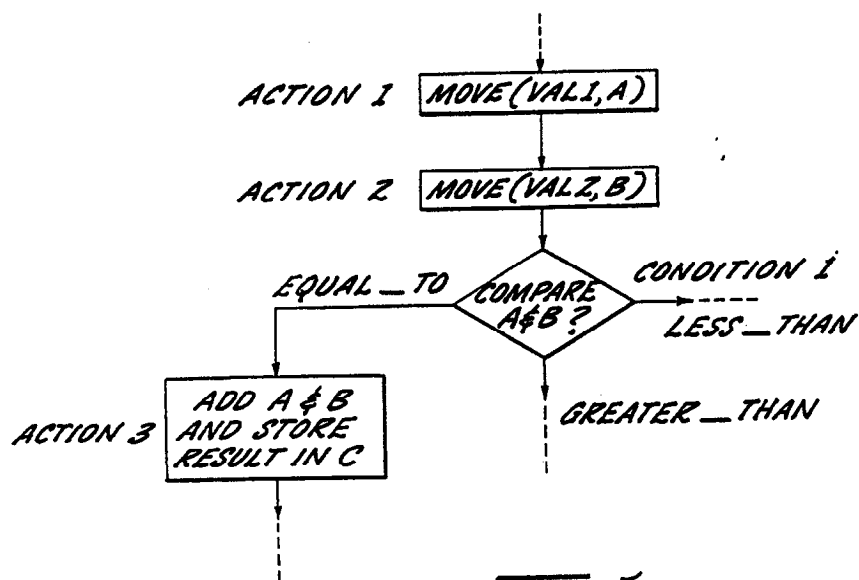
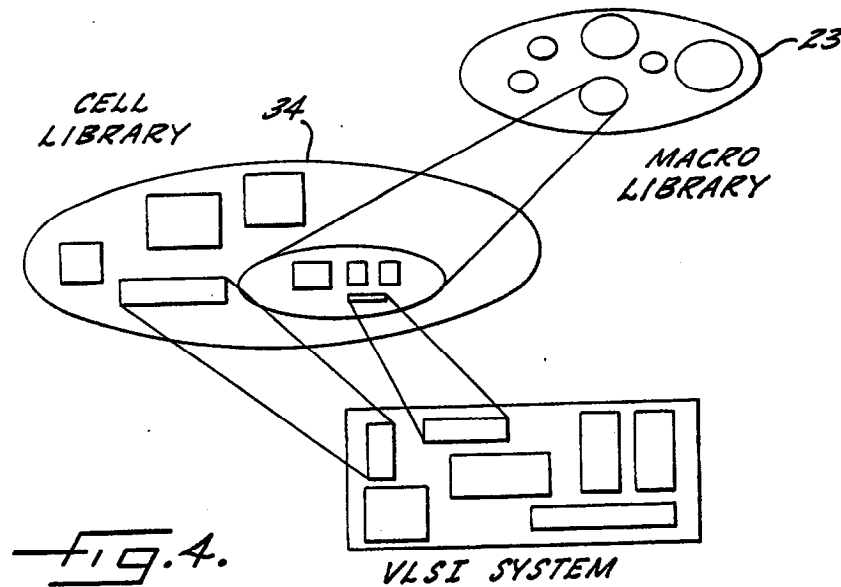
U.S. Patent

May 1, 1990

Sheet 2 of 12

4,922,432

FIG. 2.FIG. 3.



U.S. Patent

May 1, 1990

Sheet 4 of 12

4,922,432

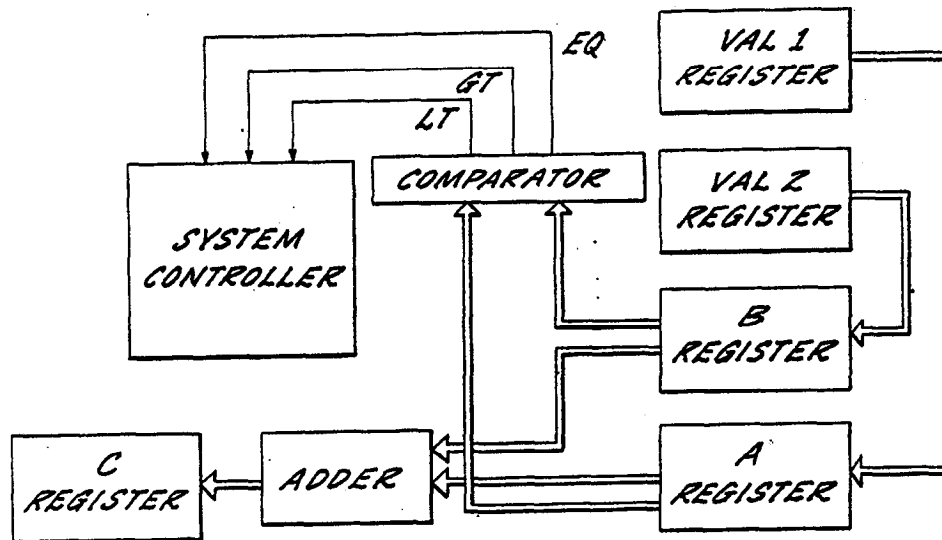


FIG. 6.

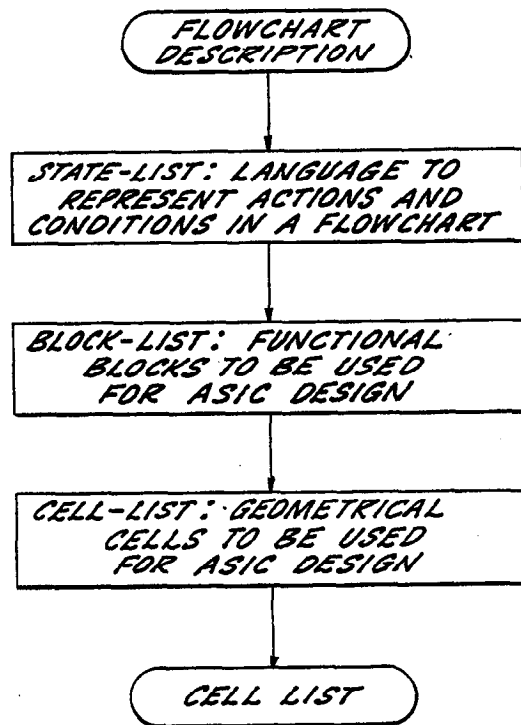


FIG. 9.

U.S. Patent

May 1, 1990

Sheet 5 of 12

4,922,432

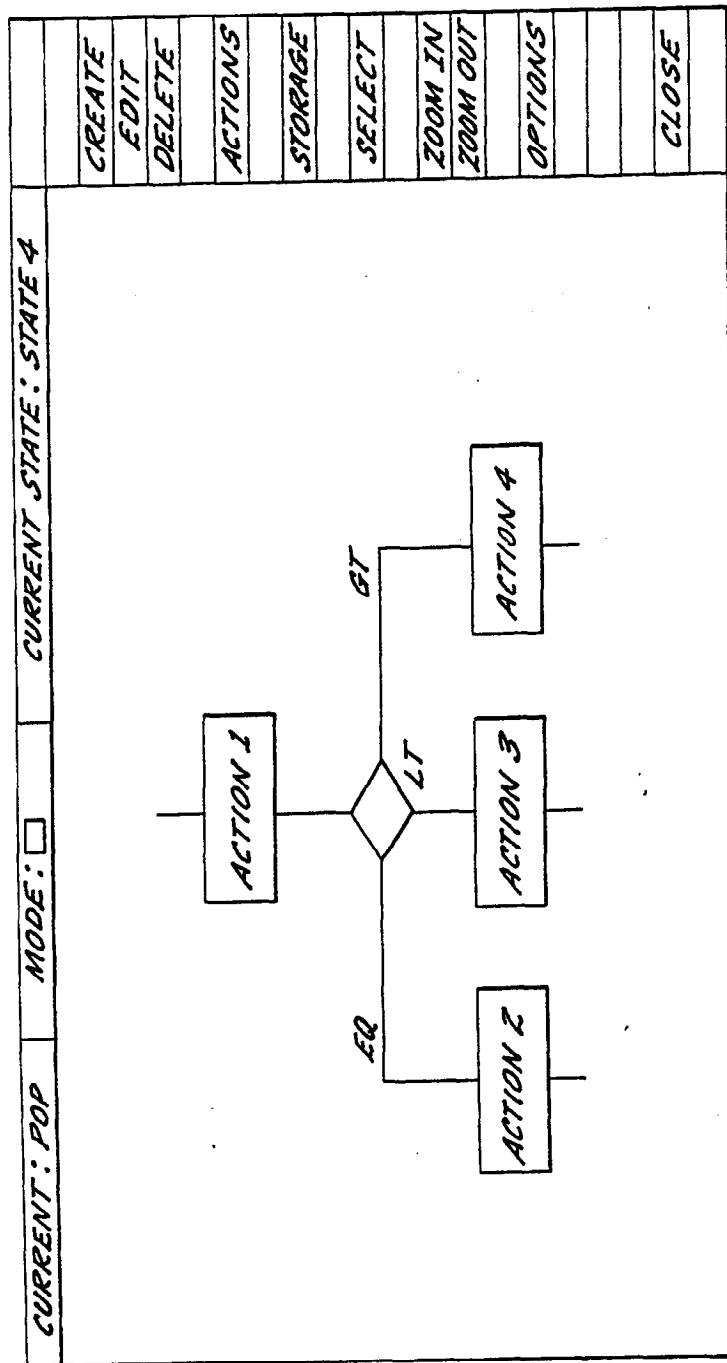


FIG. 7.

EDIT DATA	SET BREAKS	STEP	HISTORY ON	CANCEL
SHOW DATA	CLEAR BREAKS	EXECUTE	DETAIL	HELP
SET STATE	SHOW BREAKS	STOP		CLOSE

*** READY ***

Fig. 8.

U.S. Patent

May 1, 1990

Sheet 7 of 12

4,922,432

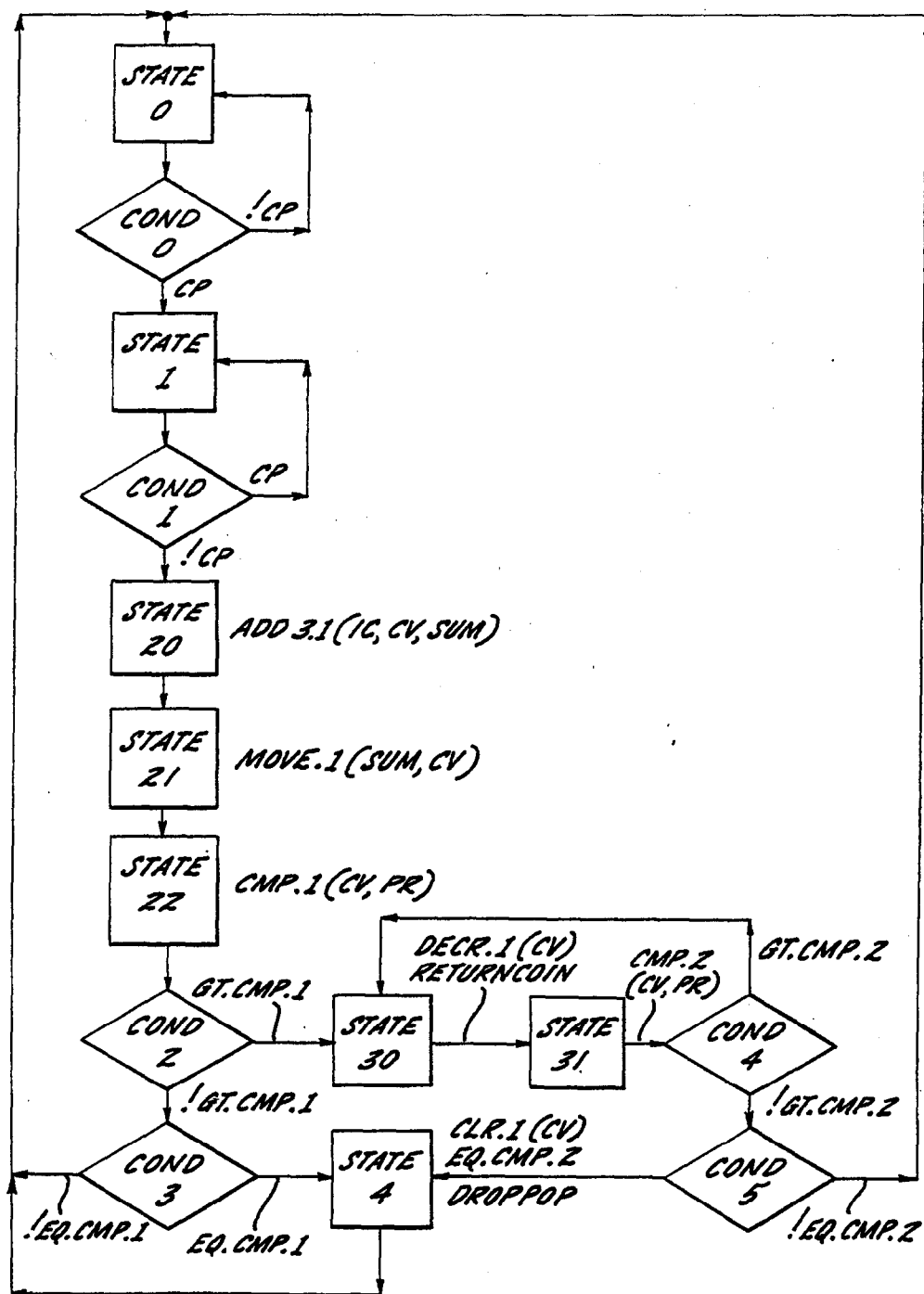


FIG. 10.

U.S. Patent

May 1, 1990

Sheet 8 of 12

4,922,432

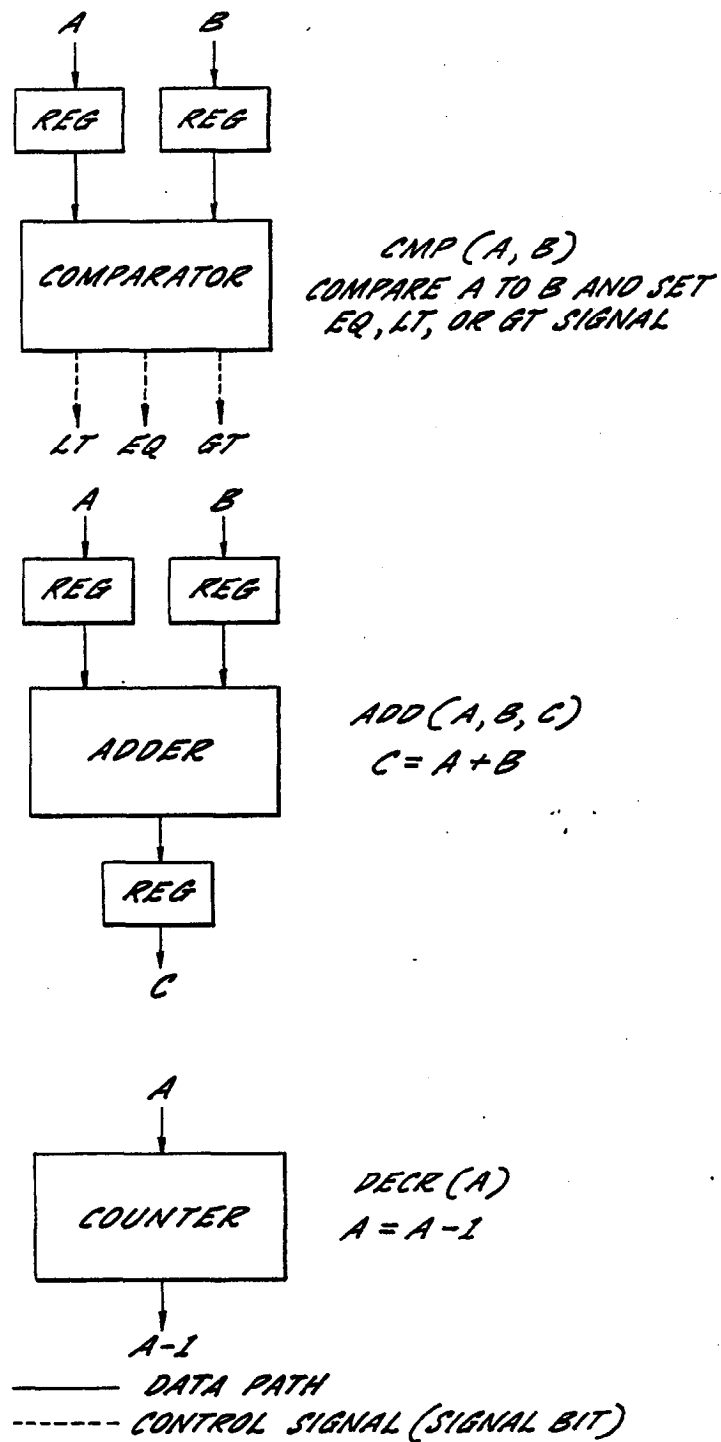


FIG. 11.

U.S. Patent

May 1, 1990

Sheet 9 of 12

4,922,432

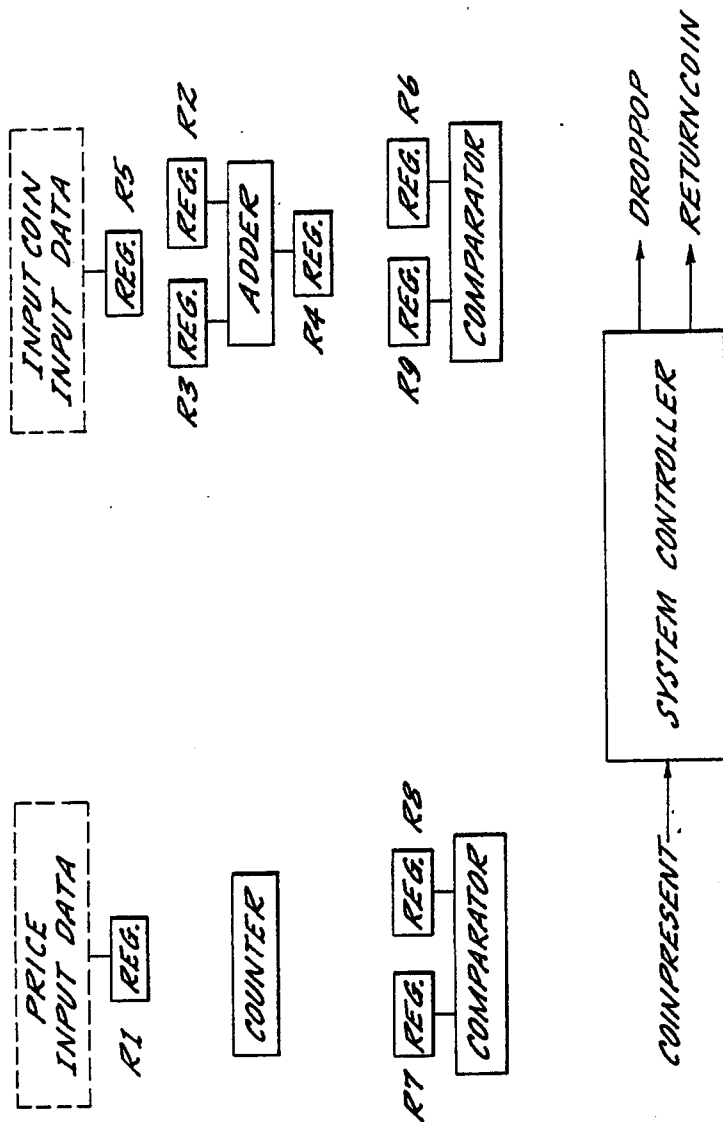


FIG. 12.

U.S. Patent

May 1, 1990

Sheet 10 of 12

4,922,432

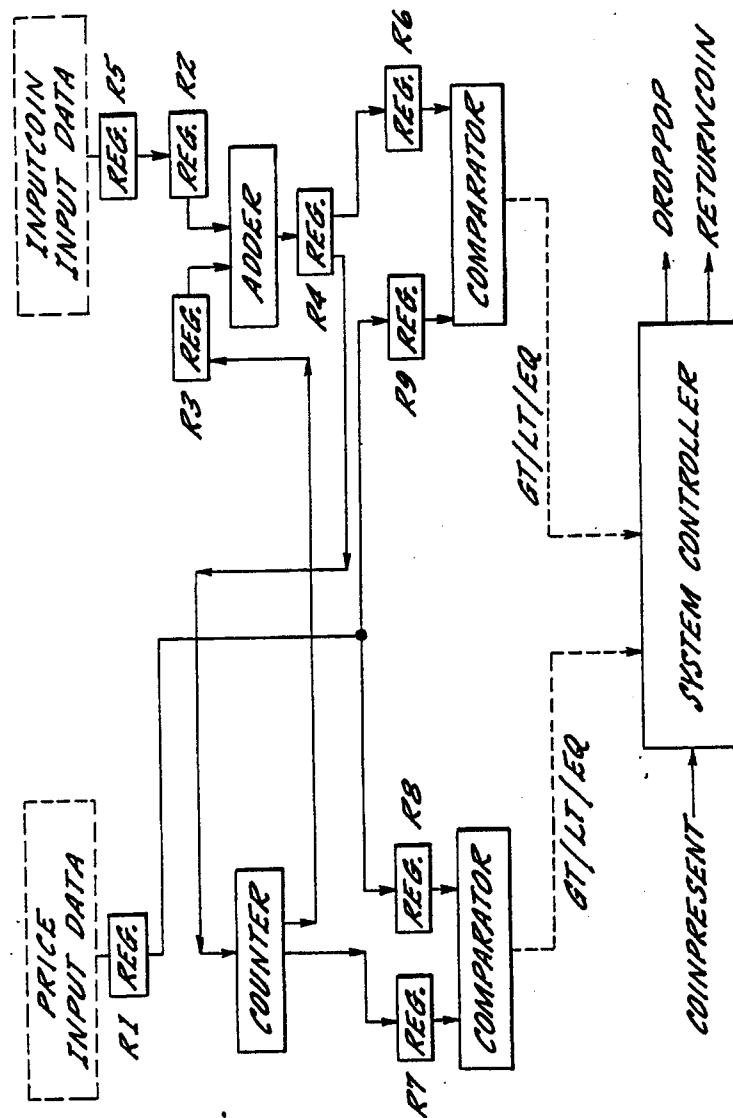


FIG. 13.

U.S. Patent

May 1, 1990

Sheet 11 of 12

4,922,432

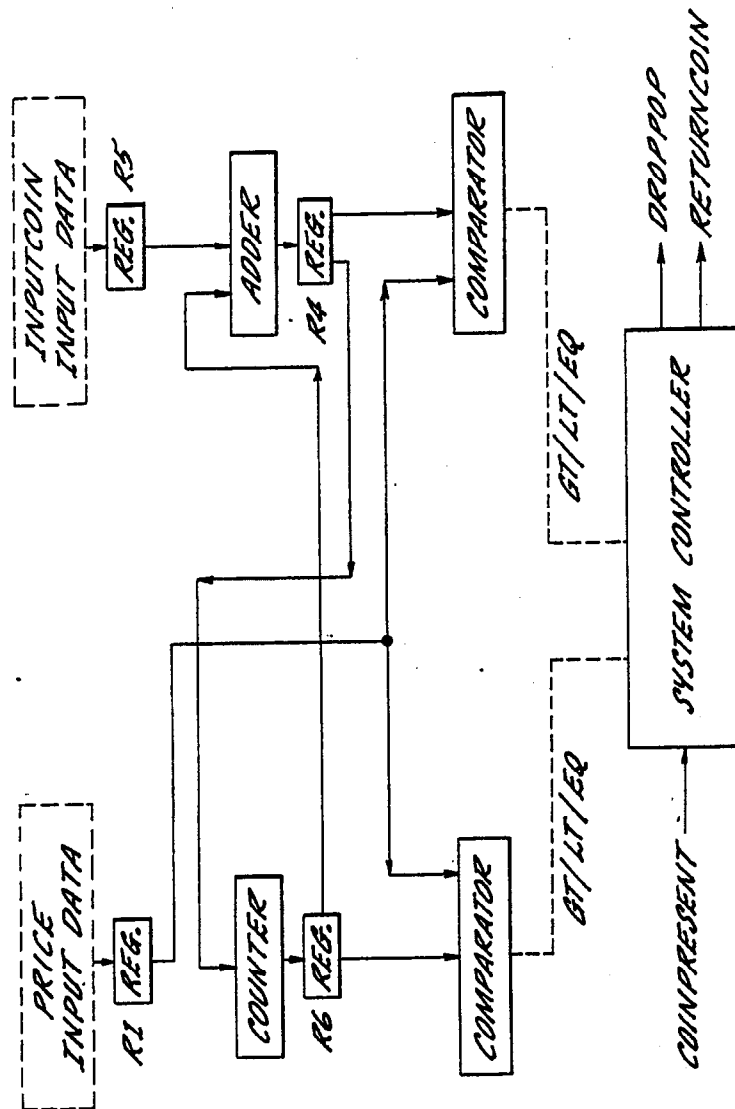


FIG. 14.

U.S. Patent

May 1, 1990

Sheet 12 of 12

4,922,432

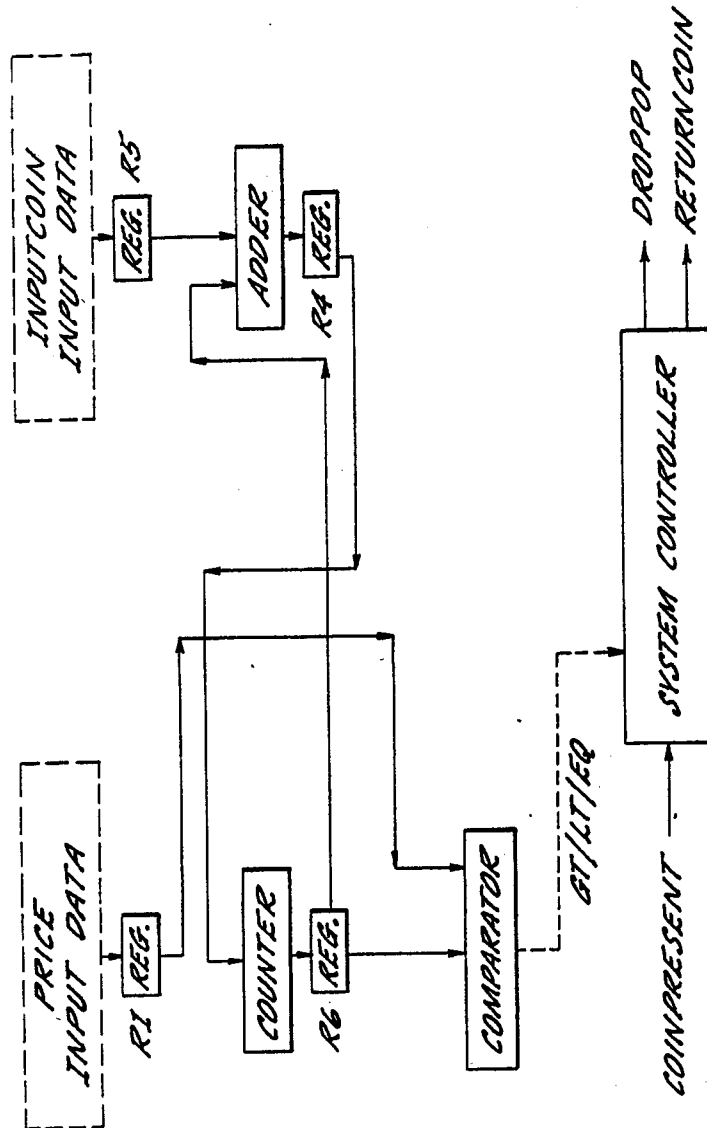


FIG. 15.

1

KNOWLEDGE BASED METHOD AND APPARATUS FOR DESIGNING INTEGRATED CIRCUITS USING FUNCTIONAL SPECIFICATIONS

FIELD AND BACKGROUND OF THE INVENTION

This invention relates to the design of integrated circuits, and more particularly relates to a computer-aided method and apparatus for designing integrated circuits.

An application specific integrated circuit (ASIC) is an integrated circuit chip designed to perform a specific function, as distinguished from standard, general purpose integrated circuit chips, such as microprocessors, memory chips, etc. A highly skilled design engineer having specialized knowledge in VLSI circuit design is ordinarily required to design a ASIC. In the design process, the VLSI design engineer will consider the particular objectives to be accomplished and tasks to be performed by the integrated circuit and will create structural level design specifications which define the various hardware components required to perform the desired function, as well as the interconnection requirements between these components. A system controller must also be designed for synchronizing the operations of these components. This requires an extensive and all encompassing knowledge of the various hardware components required to achieve the desired objectives, as well as their interconnection requirements, signal level compatibility, timing compatibility, physical layout, etc. At each design step, the designer must do tedious analysis. The design specifications created by the VLSI design engineer may, for example, be in the form of circuit schematics, parameters or specialized hardware description languages (HDLs).

From the structural level design specifications, the description of the hardware components and interconnections is converted to a physical chip layout level description which describes the actual topological characteristics of the integrated circuit chip. This physical chip layout level description provides the mask data needed for fabricating the chip.

Due to the tremendous advances in very large scale integration (VLSI) technology, highly complex circuit systems are being built on a single chip. With their complexity and the demand to design custom chips at a faster rate, in large quantities, and for an ever increasing number of specific applications, computer-aided design (CAD) techniques need to be used. CAD techniques have been used with success in design and verification of integrated circuits, at both the structural level and at the physical layout level. For example, CAD systems have been developed for assisting in converting VLSI structural level descriptions of integrated circuits into the physical layout level topological mask data required for actually producing the chip. Although the presently available computer-aided design systems greatly facilitate the design process, the current practice still requires highly skilled VLSI design engineers to create the necessary structural level hardware descriptions.

There is only a small number of VLSI designers who possess the highly specialized skills needed to create structural level integrated circuit hardware descriptions. Even with the assistance of available VLSI CAD tools, the design process is time consuming and the probability of error is also high because of human in-

2

volvements. There is a very significant need for a better and more cost effective way to design custom integrated circuits.

SUMMARY OF THE INVENTION

In accordance with the present invention a CAD (computer-aided design) system and method is provided which enables a user to define the functional requirements for a desired target integrated circuit, using an easily understood functional architecture independent level representation, and which generates therefrom the detailed information needed for directly producing an application specific integrated circuit (ASIC) to carry out those specific functions. Thus, the present invention, for the first time, opens the possibility for the design and production of ASICs by designers, engineers and technicians who may not possess the specialized expert knowledge of a highly skilled VLSI design engineer.

The functional architecture independent specifications of the desired ASIC can be defined in a suitable manner, such as in list form or preferably in a flowchart format. The flowchart is a highly effective means of describing a sequence of logical operations, and is well understood by software and hardware designers of varying levels of expertise and training. From the flowchart (or other functional specifications), the system and method of the present invention translates the functional architecture independent specifications into structural an architecture specific level definition of an integrated circuit, which can be used directly to produce the ASIC. The structural level definition includes a list of the integrated circuit hardware cells needed to achieve the functional specifications. These cells are selected from a cell library of previously designed hardware cells of various functions and technical specifications. The system also generates data paths among the selected hardware cells. In addition, the present invention generates a system controller and control paths for the selected integrated circuit hardware cells. The list of hardware cells and their interconnection requirements may be represented in the form of a netlist. From the netlist it is possible using either known manual techniques or existing VLSI CAD layout systems to generate the detailed chip level geometrical information (e.g. mask data) required to produce the particular application specific integrated circuit in chip form.

The preferred embodiment of the system and method of the present invention which is described more fully hereinafter is referred to as a Knowledge Based Silicon Compiler (KBSC). The KBSC is an ASIC design methodology based upon artificial intelligence and expert systems technology. The user interface of KBSC is a flowchart editor which allows the designer to represent VLSI systems in the form of a flowchart. The KBSC utilizes a knowledge based expert system, with a knowledge base extracted from expert ASIC designers with a high level of expertise in VLSI design to generate from the flowchart a netlist which describes the selected hardware cells and their interconnection requirements.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be better understood by reference to the detailed description which follows, taken in connection with the accompanying drawings, in which

4,922,432

3

FIG. 1a illustrates a functional level design representation of a portion of a desired target circuit, shown in the form of a flowchart;

FIG. 1b illustrates a structural level design representation of an integrated circuit;

FIG. 1c illustrates a design representation of a circuit at a physical layout level, such as would be utilized in the fabrication of an integrated circuit chip;

FIG. 2 is a block schematic diagram showing how integrated circuit mask data is created from flowchart descriptions by the KBSC system of the present invention;

FIG. 3 is a somewhat more detailed schematic illustration showing the primary components of the KBSC system;

FIG. 4 is a schematic illustration showing how the ASIC design system of the present invention draws upon selected predefined integrated circuit hardware cells from a cell library;

FIG. 5 is an example flowchart defining a sequence of functional operations to be performed by an integrated circuit;

FIG. 6 is a structural representation showing the hardware blocks and interconnection requirements for the integrated circuit defined in FIG. 5;

FIG. 7 is an illustration of the flowchart editor window;

FIG. 8 is an illustration of the flowchart simulator window;

FIG. 9 is an illustration of the steps involved in cell list generation;

FIG. 10 is an example flowchart for a vending machine system;

FIG. 11 illustrates the hardware components which correspond to each of the three macros used in the flowchart of FIG. 10;

FIG. 12 is an initial block diagram showing the hardware components for an integrated circuit as defined in the flowchart of FIG. 10;

FIG. 13 is a block diagram corresponding to FIG. 12 showing the interconnections between blocks;

FIG. 14 is a block diagram corresponding to FIG. 13 after register optimization; and

FIG. 15 is a block diagram corresponding to FIG. 14 after further optimization.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

FIGS. 1a, 1b and 1c illustrate three different levels of representing the design of an integrated circuit. FIG. 1a shows a functional (or behavioral) representation architecture independent in the form of a flowchart. A flowchart is a graphic representation of an algorithm and consists of two kinds of blocks or states, namely actions and conditions (decisions). Actions are conventionally represented in the flowchart by a rectangle or box, and conditions are represented by a diamond. Transitions between actions and conditions are represented by lines with arrows. FIG. 1b illustrates a structural (or logic) level representation of an integrated circuit. In this representation, blocks are used to represent integrated architecture specific circuit hardware components for performing various functions, and the lines interconnecting the blocks represent paths for the flow of data or control signals between the blocks. The blocks may, for example, represent hardware components such as adders, comparators, registers, system controllers, etc. FIG. 1c illustrates a physical layout level representation

4

of an integrated circuit design, which provides the detailed mask data necessary to actually manufacture the devices and conductors which together comprise integrated circuit.

As noted earlier, the design of an integrated circuit at the structural level requires a design engineer with highly specialized skills and expertise in VLSI design. In the KBSC system of the present invention, however, integrated circuits can be designed at a functional level because the expertise in VLSI design is provided and applied by the invention. Allowing the designer to work with flowcharts instead of logic circuit schematics simplifies the task of designing custom integrated circuits, making it quicker, less expensive and more reliable. The designer deals with an algorithm using simple flowcharts at an architecture independent functional (behavioral) level, and needs to know only the necessary logical steps to complete a task, rather than the specific means for accomplishing the task. Designing with flowcharts requires less work in testing because flowcharts allow the designer to work much closer to the algorithm. On the other hand, previously existing VLSI design tools require the designer to represent an algorithm with complex circuit schematics at a structural level, therefore requiring more work in testing. Circuit schematics make it harder for the designer to cope with the algorithm function which needs to be incorporated into the target design because they intermix the hardware and functional considerations. Using flowcharts to design custom integrated circuits will allow a large number of system designers to access VLSI technology, where previously only a small number of designers had the knowledge and skills to create the necessary structural level hardware descriptions.

The overall system flow is illustrated in FIG. 2. The user enters the functional specifications of the circuit into the knowledge based silicon compiler (KBSC) in the form of a flowchart 11. The KBSC 10 then generates a netlist 15 from the flowchart. The netlist 15 includes a custom generated system controller, all other hardware cells required to implement the necessary operations, and interconnection information for connecting the hardware cells and the system controller. The netlist can be used as input to any existing VLSI layout and routing tool 16 to create mask data 18 for geometrical layout.

System Overview

The primary elements or modules which comprise the KBSC system are shown in FIG. 3. In the embodiment illustrated and described herein, these elements or modules are in the form of software programs, although persons skilled in the appropriate art will recognize that these elements can easily be embodied in other forms, such as in hardware.

Referring more particularly to FIG. 3, it will be seen that the KBSC system 10 includes a program 20 called EDSIM, which comprises a flowchart editor 21 for creating and editing flowcharts and a flowchart simulator 22 for simulation and verification of flowcharts. Actions to be performed by each of the rectangles represented in the flowchart are selected from a macro library 23. A program 30 called PSCS (path synthesizer and cell selector) includes a data and control path synthesizer module 31, which is a knowledge based system for data and control path synthesis. PSCS also includes a cell selector 32 for selecting the cells required for system design. The cell selector 32 selects from a cell

library 34 of previously designed hardware cells the appropriate cell or cells required to perform each action and condition represented in the flowchart. A controller generator 33 generates a custom designed system controller for controlling the operations of the other hardware cells. The knowledge base 35 contains ASIC design expert knowledge required for data path synthesis and cell selection. Thus, with a functional flowchart input, PSCS generates a system controller, selects all other hardware cells, generates data and control paths, and generates a netlist describing all of this design information.

The KBSC system employs a hierarchal cell selection ASIC design approach, as is illustrated in FIG. 4. Rather than generating every required hardware cell from scratch, the system draws upon a cell library 34 of previously designed, tested and proven hardware cells of various types and of various functional capabilities with a given type. The macro library 23 contains a set of macros defining various actions which can be specified in the flowchart. For each macro function in the macro library 23 there may be several hardware cells in the cell library 34 of differing geometry and characteristics capable of performing the specified function. Using a rule based expert system with a knowledge base 35 extracted from expert ASIC designers, the KBSC system selects from the cell library 34 the optimum cell for carrying out the desired function.

Referring again to FIG. 3, the cells selected by the cell selector 32, the controller information generated by the controller generator 33 and the data and control paths generated by the data/control path synthesizer 31 are all utilized by the PSCS program 30 to generate the netlist 15. The netlist is a list which identifies each block in the circuit and the interconnections between the respective inputs and outputs of each block. The netlist provides all the necessary information required to produce the integrated circuit. Computer-aided design systems for cell placement and routing are commercially available which will receive netlist data as input and will lay out the respective cells in the chip, generate the necessary routing, and produce mask data which can be directly used by a chip foundry in the fabrication of integrated circuits.

System Requirements

The KBSC system can be operated on a suitable programmed general purpose digital computer. By way of example, one embodiment of the system is operated in a work station environment such as Sun3 and VAXStation-II/GPX Running UNIX Operating System and X Window Manager. The work station requires a minimum of 8 megabytes of main storage and 20 megabytes of hard disk space. The monitor used is a color screen with 8-bit planes. The software uses C programming language and INGRES relational data base.

The human interface is mainly done by the use of pop up menus, buttons, and a special purpose command language. The permanent data of the integrated circuit design are stored in the data base for easy retrieval and update. Main memory stores the next data temporarily, executable code, design data (flowchart, logic, etc.), data base (cell library), and knowledge base. The CPU performs the main tasks of creating and simulating flowcharts and the automatic synthesis of the design.

Flowchart Example

To describe the mapping of a flowchart to a netlist, consider an example flowchart shown in FIG. 5, which is of part of a larger overall system. In this illustrative flowchart, two variables, VAL1 and VAL2 are compared and if they are equal, they are added together. In this instance, the first action (Action 1) involves moving the value of variable VAL1 to register A. The second action comprises moving the value of variable VAL2 to register B. Condition 1 comprises comparing the values in registers A and B. Action 3 comprises adding the values of registers A and B and storing the result in register C.

In producing an integrated circuit to carry out the function defined in FIG. 5, the KBSC maps the flowchart description of the behavior of the system to interconnection requirements between hardware cells. The hardware cells are controlled by a system controller which generates all control signals. There are two types of variables involved in a system controller:

(1) Input variables: These are generated by hardware cells, and/or are external input to the controller. These correspond to conditions in the flowchart.

(2) Output variables: These are generated by the system controller and correspond to actions in the flowchart.

FIG. 6 illustrates the results of mapping the flowchart of FIG. 5 onto hardware cells. The actions and the conditions in the flowchart are used for cell selection and data and control path synthesis. The VAL1 register and VAL2 register and the data paths leading therefrom have already been allocated in actions occurring before Action 1 in our example. Action 1 causes generation of the data register A. Similarly, Action 2 causes the allocation of data register B. The comparator is allocated as a result of the comparison operation in Condition 1. The comparison operation is accomplished by (1) selecting a comparator cell, (2) mapping the inputs of the comparator cell to registers A and B, (3) generating data paths to connect the comparator with the registers A and B and (4) generating input variables corresponding to equal to, greater than, and less than for the system controller. Similarly the add operation in Action 3 causes selection of the adder cell, mapping of the adder parameters to the registers and creating the data paths.

Following this methodology, a block list can be generated for a given flowchart. This block list consists of a system controller and as many other blocks as may be required for performing the necessary operations. The blocks are connected with data paths, and the blocks are controlled by the system controller through control paths. These blocks can be mapped to the cells selected from a cell library to produce a cell list.

Interactive Flowchart Editor and Simulator

The creation and verification of the flowchart is the first step in the VLSI design methodology. The translation from an algorithm to an equivalent flowchart is performed with the Flowchart Editor 21 (FIG. 3). The verification of the edited flowchart is performed by the Flowchart Simulator 22. The Flowchart Editor and Simulator are integrated into one working environment for interactive flowchart editing, with a designer friendly interface.

EDSIM is the program which contains the Flowchart Editor 21 and the Flowchart Simulator 22. It also provides functions such as loading and saving flow-

charts. EDSIM will generate an intermediate file, called a statelist, for each flowchart. This file is then used by the PSCS program 30 to generate a netlist.

Flowchart Editor

The Flowchart Editor 21 is a software module used for displaying, creating, and editing the flowchart. This module is controlled through the flowchart editing window illustrated in FIG. 7. Along with editing functions the Flowchart Editor also provides checking of design errors.

The following is a description of the operations of the Flowchart Editor. The main editing functions include, create, edit, and delete states, conditions, and transitions. The create operation allows the designer to add a new state, condition, or transitions to a flowchart. Edit allows the designer to change the position of a state, condition or transition, and delete allows the designer to remove a state, condition or transition from the current flowchart. States which contain actions are represented by boxes, conditions are represented by diamonds, and transitions are represented by lines with arrows showing the direction of the transition.

Edit actions allows the designer to assign actions to each box. These actions are made up of macro names and arguments. An example of arguments is the setting and clearing of external signals. A list of basic macros available in the macro library 23 is shown in Table 1.

TABLE 1

Macro	Description
ADD (A,B,C)	$C = A + B$
SUB (A,B,C)	$C = A - B$
MULT (A,B,C)	$C = A * B$
DIV (A,B,C)	$C = A \text{ div } B$
DECR (A)	$A = A - 1$
INCR (A)	$A = A + 1$
CLR (A)	$A = 0$
REG (A,B)	$B = A$
CMP (A,B)	Compare A to B and set EQ,LT,GT signals
CMP0 (A)	Compare A to 0 and set EQ,LT,GT signals
NEGATE (A)	$A = \text{NOT } (A)$
MOD (A,B,C)	$C = A \text{ Modulus } B$
POW (A,B,C)	$C = A^B$
DC2 (A,S1,S2,S3,S4)	Decode A into S1,S2,S3,S4
EC2 (S1,S2,S3,S4,A)	Encode S1,S2,S3,S4 into A
MOVE (A,B)	$B = A$
CALL sub-flowchart (A,B,...)	Call a sub-flowchart. Pass A,B...
START (A,B,...)	Beginning state of a sub-flowchart
STOP (A,B,...)	Ending state of a sub-flowchart

The Flowchart Editor also provides a graphical display of the flowchart as the Flowchart Simulator simulates the flowchart. This graphical display consists of boxes, diamonds, and lines as shown in FIG. 7. All are drawn on the screen and look like a traditional flowchart. By displaying the flowchart on the screen during simulation it allows the designer to design and verify the flowchart at the same time.

Flowchart Simulator

The Flowchart Simulator 22 is a software module used for simulating flowcharts. This module is controlled through the simulator window illustrated in FIG. 8. The Flowchart Simulator simulates the transitions between states and conditions in a flowchart. The following is a list of the operations of the Flowchart Simulator:
edit data—Change the value of a register or memory.

set state—Set the next state to be simulated.
set detail or summary display—Display summary or detail information during simulation.
set breaks—Set a breakpoint.
clear breaks—Clear all breakpoints.
show breaks—Display current breakpoints.
step—Step through one transition.
execute—Execute the flowchart.
stop—Stop executing of the flowchart. history ON or history OFF—Set history recording on or off.
cancel—Cancel current operation.
help—Display help screen.
close—Close the simulator window.

The results of the simulation are displayed within the simulator window. Also the editor window will track the flowchart as it is being simulated. This tracking of the flowchart makes it easy to edit the flowchart when an error is found.

Cell Selection

The Cell Selector 32 is a knowledge based system for selecting a set of optimum cells from the cell library 34 to implement a VLSI system. The selection is based on functional descriptions in the flowchart, as specified by the macros assigned to each action represented in the flowchart. The cells selected for implementing a VLSI system depend on factors such as cell function, fabrication technology used, power limitations, time delays etc. The cell selector uses a knowledge base extracted from VLSI design experts to make the cell selection.

To design a VLSI system from a flowchart description of a user application, it is necessary to match the functions in a flowchart with cells from a cell library. This mapping needs the use of artificial intelligence techniques because the cell selection process is complicated and is done on the basis of a number of design parameters and constraints. The concept used for cell selection is analogous to that used in software compilation. In software compilation a number of subroutines are linked from libraries. In the design of VLSI systems, a functional macro can be mapped to library cell.

FIG. 4 illustrates the concept of hierarchical cell selection. The cell selection process is performed in two steps:

- (1) selection of functional macros
- (2) selection of geometrical cells

A set of basic macros is shown in Table 1. A macro corresponds to an action in the flowchart. As an example, consider the operation of adding A and B and storing the result in C. This function is mapped to the addition macro ADD(X, Y, Z). The flowchart editor and flowchart simulator are used to draw the rectangles, diamonds and lines of the flowchart, to assign a macro selected from the macro library 23 to each action represented in the flowchart, and to verify the functions in flowcharts. The flowchart is converted into an intermediate form (statelist) and input to the Cell Selector.

The Cell Selector uses a rule based expert system to select the appropriate cell or cells to perform each action. If the cell library has a number of cells with different geometries for performing the operation specified by the macro, then an appropriate cell can be selected on the basis of factors such as cell function, process technology used, time delay, power consumption, etc.

The knowledge base of Cell Selector 32 contains information (rules) relating to:

- (1) selection of macros
- (2) merging two macros

- (3) mapping of macros to cells
- (4) merging two cells
- (5) error diagnostics

The above information is stored in the knowledge base 35 as rules.

Cell List Generation

FIG. 9 shows the cell list generation steps. The first step of cell list generation is the transformation of the flowchart description into a structure that can be used by the Cell Selector. This structure is called the statelist. The blocklist is generated from the statelist by the inference engine. The blocklist contains a list of the functional blocks to be used in the integrated circuit. Rules of the following type are applied during this stage.

- map arguments to data paths
- map actions to macros
- connect these blocks

Rules also provide for optimization and error diagnostics at this level.

The cell selector maps the blocks to cells selected from the cell library 34. It selects an optimum cell for a block. This involves the formulation of rules for selecting appropriate cells from the cell library. Four types of information are stored for each cell. These are:

- (1) functional level information: description of the cell at the register transfer level.
- (2) logic level information: description in terms of flip-flops and gates.
- (3) circuit level information: description at the transistor level.
- (4) Layout level information: geometrical mask level specification.

The attributes of a cell are:

- cell name
- description
- function
- width
- height
- status
- technology
- minimum delay
- typical delay
- maximum delay
- power
- file
- designer
- date
- comment
- inspector

In the cell selection process, the above information can be used. Some parameters that can be used to map macros to cells are:

- (1) name of macro
- (2) function to be performed
- (3) complexity of the chip
- (4) fabrication technology
- (5) delay time allowed
- (6) power consumption
- (7) bit size of macro data paths

Netlist Generation

The netlist is generated after the cells have been selected by PSCS. PSCS also uses the macro definitions for connecting the cell terminals to other cells. PSCS uses the state-to-state transition information from an intermediate form representation of a flowchart (i.e. the

statelist) to generate a netlist. PSCS contains the following knowledge for netlist generation:

- (1) Data path synthesis
- (2) Data path optimization
- (3) Macro definitions
- (4) Cell library
- (5) Error detection and correction

The above information is stored in the knowledge base 35 as rules. Knowledge engineers help in the formulation of these rules from ASIC design experts. The macro library 23 and the cell library 34 are stored in a database of KBSC.

A number of operations are performed by PSCS. The following is a top level description of PSCS operations:

- (1) Read the flowchart intermediate file and build a statelist.
- (2) current_context=START
- (3) Start the inference engine and load the current context rules.
- (4) Perform one of the following operations depending upon current_context:
 - (a) Modify the statelist for correct implementation.
 - (b) Create blocklist, macrolist and data paths.
 - (c) Optimize blocklist and datapath list and perform error checks.
 - (d) Convert blocks to cells.
 - (e) Optimize cell list and perform error checks.
 - (f) Generate netlist.
 - (g) Optimize netlist and perform error checks and upon completion Goto 7.
- (5) If current_context has changed, load new context rules.
- (6) Goto 4.
- (7) Output netlist file and stf files and Stop.

In the following sections, operations mentioned in step 4 are described. The Rule Language and PSCS display are also described.

Rule Language

The rule language of PSCS is designed to be declarative and to facilitate rule editing. In order to make the expert understand the structure of the knowledge base, the rule language provides means for knowledge representation. This will enable the format of data structures to be stated in the rule base, which will enable the expert to refer to them and understand the various structures used by the system. For example, the expert can analyze the structure of wire and determine its components. The expert can then refer these components into rules. If a new object has to be defined, then the expert can declare a new structure and modify some existing structure to link to this new structure. In this way, the growth of the data structures can be visualized better by the expert. This in turn helps the designer to update and append rules.

The following features are included in the rule language:

- (i) Knowledge representation in the form of a record structure.
- (ii) Conditional expressions in the antecedent of a rule.
- (iii) Facility to create and destroy structure in rule actions.
- (iv) The assignment statement in the action of a rule.
- (v) Facility for input and output in rule actions.
- (vi) Provide facility to invoke C functions from rule actions.

The rule format to be used is as follows:

The rule format to be used is as follows:		
Rule	<number>	<context>
If {	<if-clause>	
}		
Then {	<then-clause>	
}		
where	<number>	rule number
	<context>	context in which this rule is active
	<if-clause>	the condition part of the rule
	<then-clause>	the action part of the rule

Inference Strategy

The inference strategy is based on a fast pattern matching algorithm. The rules are stored in a network and the requirement to iterate through the rules is avoided. This speeds up the execution. The conflict resolution strategy to be used is based on the following:

- (1) The rule containing the most recent data is selected.
- (2) The rule which has the most complex condition is selected.
- (3) The rule declared first is selected.

Rule Editor

PSCS provides an interactive rule editor which enables the expert to update the rule set. The rules are stored in a database so that editing capabilities of the database package can be used for rule editing. To perform this operation the expert needs to be familiar with the various knowledge structures and the inferencing process. If this is not possible, then the help of a knowledge engineer is needed.

PSCS provides a menu from which various options can be set. Mechanisms are provided for setting various debugging flags and display options, and for the overall control of PSCS.

Facility is provided to save and display the blocklist created by the user. The blocklist configuration created by the user can be saved in a file and later be printed with a plotter. Also the PSCS display can be reset to restart the display process.

PSCS Example Rules:		
Rule 1	IF	no blocks exist
	THEN	generate a system controller.
Rule 2	IF	a state exists which has a macro AND this macro has not been mapped to a block
	THEN	find a corresponding macro in the library and generate a block for this macro.
Rule 3	IF	there is a transition between two states AND there are macros in these states using the same argument
	THEN	make a connection from a register corresponding to the first macro to another register corresponding to the second macro.
Rule 4	IF	a register has only a single connection from another register
	THEN	combine these registers into a single register.
Rule 5	IF	there are two comparators AND input data widths are of the same size AND

-continued

PSCS Example Rules:		
		one input of these is same AND the outputs of the comparators are used to perform the same operation. combine these comparators into a single comparator.
	THEN	
Rule 6	IF	there is a data without a register
	THEN	allocate a register for this data.
Rule 7	IF	all the blocks have been interconnected AND a block has a few terminals not connected
	THEN	remove the block and its terminals, or issue an error message.
Rule 8	IF	memory is to be used, but a block has not been created for it
	THEN	create a memory block with data, address, read and write data and control terminals.
Rule 9	IF	a register has a single connection to a counter
	THEN	combine the register and the counter; remove the register and its terminals.
Rule 10	IF	there are connections to a terminal of a block from many different blocks
	THEN	insert a multiplexor; remove the connections to the terminals and connect them to the input of the multiplexor; connect the output of the multiplexor to the input of the block.

Additional rules address the following points:
remove cell(s) that can be replaced by using the outputs of other cell(s)
reduce multiplexor trees
use fan-out from the cells, etc.

Soft Drink Vending Machine Controller Design Example

The following example illustrates how the previously described features of the present invention are employed in the design of an application specific integrated circuit (ASIC). In this illustrative example the ASIC is designed for use as a vending machine controller. The vending machine controller receives a signal each time a coin has been deposited in a coin receiver. The coin value is recorded and when coins totalling the correct amount are received, the controller generates a signal to dispense a soft drink. When coins totalling more than the cost of the soft drink are received, the controller dispenses change in the correct amount.

This vending machine controller example is patterned after a textbook example used in teaching digital system controller design. See Fletcher, William I., *An Engineering Approach to Digital Design*, Prentice-Hall, Inc., pp. 491-505. Reference may be made to this textbook example for a more complete explanation of this vending machine controller requirements, and for an understanding and appreciation of the complex design procedures prior to the present invention for designing the hardware components for a controller.

FIG. 10 illustrates a flowchart for the vending machine controller system. This flowchart would be entered into the KBSC system by the user through the flowchart editor. Briefly reviewing the flowchart, the controller receives a coin present signal when a coin is received in the coin receiver. State0 and cond0 define a waiting state awaiting deposit of a coin. The symbol CP represents "coin present" and the symbol !CP repre-

13

sents "coin not present". State1 and cond1 determine when the coin has cleared the coin receiver. At state20, after receipt of a coin, the macro instruction ADD3.1 (lc, cv, sum) instructs the system to add lc (last coin) and cv (coin value) and store the result as sum. The macro instruction associated with state21 moves the value in the register sum to cv. The macro CMP.1 at state22 compares the value of cv with PR (price of soft drink) and returns signals EQ, GT and LT. The condition cond2 tests the result of the compare operation CMP.1. If the result is "not greater than" (!GT.CMP.1), then the condition cond3 tests to see whether the result is "equal" (EQ.CMP.1). If the result is "not equal" (!EQ.CMP.1), then control is returned to state0 awaiting the deposit of another coin. If cond3 is EQ, then state4 generates a control signal to dispense a soft drink (droppop) and the macro instruction CLR.1(cv) resets cv to zero awaiting another customer.

If the total coins deposited exceed the price, then state30 produces the action "returncoin". Additionally, the macro DECR.1 (cv) reduces the value of cv by the amount of the returned coin. At state31 cv and PR are again compared. If cv is still greater than PR, then control passes to state30 for return of another coin. The condition cond5 tests whether the result of CMP.2 is EQ and will result in either dispensing a drink (droppop) true or branching to state0 awaiting deposit of another coin. The macros associated with the states shown in FIG. 10 correspond to those defined in Table 1 above and define the particular actions which are to be performed at the respective states.

Appendix A shows the intermediate file or "statelist" produced from the flowchart of FIG. 10. This statelist is produced as output from the EDSIM program 20 and is used as input to the PSCS program 30 (FIG. 3).

FIG. 11 illustrates for each of the macros used in the flowchart of FIG. 10, the corresponding hardware blocks. It will be seen that the comparison macro CMP (A,B) results in the generation of a register for storing value A, a register for storing value B, and a comparator block and also produces control paths to the system controller for the EQ, LT, and GT signals generated as a result of the comparison operation. The addition macro ADD (A,B,C) results in the generation of a register for each of the input values A and B, a register for the output value C, and in the generation of an adder block. The macro DECR (A) results in the generation of a counter block. The PSCS program 30 maps each of the macros used in the flowchart of FIG. 10 to the corresponding hardware components results in the generation of the hardware blocks shown in FIG. 12. In generating the illustrated blocks, the PSCS program 30 relied upon rules 1 and 2 of the above listed example rules.

FIG. 13 illustrates the interconnection of the block of FIG. 12 with data paths and control paths. Rule 3 was used by the data/control path synthesizer program 31 in mapping the data and control paths.

FIG. 14 shows the result of optimizing the circuit by applying rule 4 to eliminate redundant registers. As a result of application of this rule, the registers R2, R3, R7, R8, and R9 in FIG. 13 were removed. FIG. 15 shows the block diagram after further optimization in which redundant comparators are consolidated. This optimization is achieved in the PSCS program 30 by application of rule 5.

Having now defined the system controller block, the other necessary hardware blocks and the data and con-

14

trol paths for the integrated circuit, the PSCS program 30 now generates a netlist 15 defining these hardware components and their interconnection requirements. From this netlist the mask data for producing the integrated circuit can be directly produced using available VLSI CAD tools.

```
name rpop;
data path @ic<0:5>, cv<0:5>, sum<0:5>, @pr<0:5>;
{
state4 : state0;
state30 : state31;
state21 : state22;
state20 : state21;
state0 : lcp state0;
state0 : cp state1;
state1 : cp state1;
state1 : lcp state20;
state22 : GT.CMP.1 state30;
state22 : !GT.CMP.1*EQ.CMP.1 state4;
state22 : !GT.CMP.1*EQ.CMP.1 state0;
state31 : GT.CMP.2 state30;
state31 : !GT.CMP.2*EQ.CMP.2 state4;
state31 : !GT.CMP.2*EQ.CMP.2 state0;
state30 : returncoin;
state30 : DECR.1(cv);
state4 : droppop;
state4 : CLR.1(cv);
state31 : CMP.2(cv,pr);
state22 : CMP.1(cv,pr);
state21 : MOVE.1(sum,cv);
state20 : ADD3.1(ic,cv,sum);
}
```

That which I claimed is:

1. A computer-aided design system for designing an application specific integrated circuit directly from architecture independent functional specifications for the integrated circuit, comprising
a macro library defining a set of architecture independent operations comprised of actions and conditions;

input specification means operable by a user for defining architecture independent functional specifications for the integrated circuit, said functional specifications being comprised of a series of operations comprised of actions and conditions, said input specification means including means to permit the user to specify for each operation a macro selected from said macro library;

a cell library defining a set of available integrated circuit hardware cells for performing the available operations defined in said macro library;

cell selection means for selecting from said cell library for each macro specified by said input specification means, appropriate hardware cells for performing the operation defined by the specified macro, said cell selection means comprising an expert system including a knowledge base containing rules for selecting hardware cells from said cell library and inference engine means for selecting appropriate hardware cells from said cell library in accordance with the rules of said knowledge base; and

netlist generator means cooperating with said cell selection means for generating as output from the system a netlist defining the hardware cells which are needed to achieve the functional requirements of the integrated circuit and the connections there-between.

2. The system as defined in claim 1 wherein said input means comprises means specification for receiving user

4,922,432

15

input of a list defining the series of actions and conditions.

3. The system as defined in claim 1 additionally including mask data generator means for generating from said netlist the mask data required to produce an integrated circuit having the specified functional requirements.

4. The system as defined in claim 1 wherein said input means comprises flowchart editor means specification for creating a flowchart having elements representing said series of actions and conditions.

5. The system as defined in claim 4 additionally including flowchart simulator means for simulating the functions defined in the flowchart to enable the user to verify the operation of the integrated circuit.

6. The system as defined in claim 1 additionally including data path generator means cooperating with said cell selection means for generating data paths for the hardware cells selected by said cell selection means.

7. The system as defined in claim 6 wherein said data path generator means comprises a knowledge base containing rules for selecting data paths between hardware cells and inference engine means for selecting data paths between the hardware cells selected by said cell selection means in accordance with the rules of said knowledge base and the arguments of the specified macros.

8. The system as defined in claim 6 additionally including control generator means for generating a controller and control paths for the hardware cells selected by said cell selection means.

9. A computer-aided design system for designing an application specific integrated circuit directly from a flowchart defining architecture independent functional requirements of the integrated circuit comprising

a marco library defining a set of architecture independent operations comprised of actions and conditions;

flowchart editor means operable by a user for creating a flowchart having elements representing said architecture independent operations;

said flowchart editor means including macro specification means for permitting the user to specify for each operation represented in the flowchart a macro selected from said macro library;

a cell library defining a set of available integrated circuit hardware cells for performing the available operations defined in said macro library;

cell selection means for selecting from said cell library for each specified macro, appropriate hardware cells for performing the operation defined by the specified macro, said cell selection means comprising an expert system including a knowledge base containing rules for selecting hardware cells from said cell library and inference engine means for selecting appropriate hardware cells from said cell library in accordance with the rules of said knowledge base; and

data path generator means cooperating with said cell selection means for generating data paths for the hardware cells selected by said cell selector means, said data path generator means comprising a knowledge base containing rules for selecting data paths between hardware cells and inference engine means for selecting data paths between hardware cells selected by said cell selection means in accordance with the rules of said knowledge base and the arguments of the specified macros.

16

10. The system as defined in claim 9 additionally including control generator means for generating a controller and control paths for the hardware cells selected by said cell selection means.

11. A computer-aided design system for designing an application specific integrated circuit directly from a flowchart defining architecture independent functional requirements of the integrated circuit, comprising

flowchart editor means operable by a user for creating a flowchart having boxes representing architecture independent actions, diamonds representing architecture independent conditions, and lines with arrows representing transitions between actions and condition and including means for specifying for each box or diamond, a particular action or condition to be performed;

a cell library defining a set of available integrated circuit hardware cells for performing actions and conditions;

a knowledge base containing rules for selecting hardware cells from said cell library and for generating data and control paths for hardware cells; and

expert system means operable with said knowledge base for translating the flowchart defined by said flowchart editor means into a netlist defining the necessary hardware cells and data and control paths required in an integrated circuit having the specified functional requirements.

12. The system as defined in claim 11 including mask data generator means for generating from said netlist the mask data required to produce an integrated circuit having the specified functional requirements.

13. A computer-aided design process for designing an application specific integrated circuit which will perform a desired function comprising

storing a set of definitions of architecture independent actions and conditions;

storing data describing a set of available integrated circuit hardware cells for performing the actions and conditions defined in the stored set;

storing in an expert system knowledge base a set of rules for selecting hardware cells to perform the actions and conditions;

describing for a proposed application specific integrated circuit a series of architecture independent actions and conditions;

specifying for each described action and condition of the series one of said stored definitions which corresponds to the desired action or condition to be performed; and

selecting from said stored data for each of the specified definitions a corresponding integrated circuit hardware cell for performing the desired function of the application specific integrated circuit, said step of selecting a hardware cell comprising applying to the specified definition of the action or condition to be performed, a set of cell selection rules stored in said expert system knowledge base and generating for the selected integrated circuit hardware cells, a netlist defining the hardware cells which are needed to perform the desired function of the integrated circuit and the interconnection requirements therefor.

14. A process as defined in claim 13, including generating from the netlist the mask data required to produce an integrated circuit having the desired function.

4,922,432

17

15. A process as defined in claim 13 including the further step of generating data paths for the selected integrated circuit hardware cells.

16. A process as defined in claim 15 wherein said step of generating data paths comprises applying to the selected cells a set of data path rules stored in a knowledge base and generating the data paths therefrom.

17. A process as defined in claim 16 including the further step of generating control paths for the selected integrated circuit hardware cells.

18. A knowledge based design process for designing an application specific integrated circuit which will perform a desired function comprising

storing in a macro library a set of macros defining architecture independent actions and conditions;

storing in a cell library a set of available integrated circuit hardware cells for performing the actions and conditions;

storing in a knowledge base set of rules for selecting hardware cells from said cell library to perform the actions and conditions defined by the stored macros;

describing for a proposed application specific integrated circuit a flowchart comprised of elements representing a series of architecture independent

18

actions and conditions which carry out the function to be performed by the integrated circuit; specifying for each described action and condition of said series a macro selected from the macro library which corresponds to the action or condition; and applying rules of said knowledge base to the specified macros to select from said cell library the hardware cells required for performing the desired function of the application specific integrated circuit and generating for the selected integrated circuit hardware cells, a netlist defining the hardware cells which are needed to perform the desired function of the integrated circuit and the interconnection requirements therefor.

19. A process as defined in claim 18 also including the steps of

storing in said knowledge base a set of rules for creating data paths between hardware cells, and applying rules of said knowledge base to the specified means to create data paths for the selected hardware cells.

20. A process as defined in claim 19 also including the steps of generating a controller and generating control paths for the selected hardware cells.

* * * * *

30

35

40

45

50

55

60

65

127

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

RICOH COMPANY, LTD,

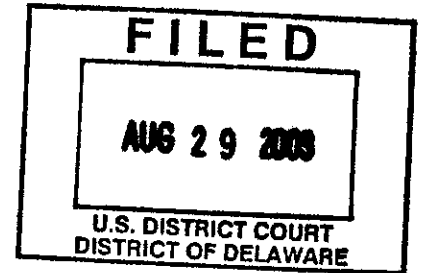
Plaintiff,

v.

Civil Action No. 03-103 GMS

AEROFLEX INCORPORATED, AMI
SEMICONDUCTOR, INC., MATROX
ELECTRONIC SYSTEMS, LTD.,
MATROX GRAPHICS, INC., MATROX
INTERNATIONAL CORP., and MATROX
TECH, INC.,

Defendants.

**MEMORANDUM AND ORDER****I. INTRODUCTION**

On January 21, 2003, the plaintiff, Ricoh Company, Ltd. ("Rico") filed the above-captioned patent infringement action against the above-named defendants. In its complaint, Ricoh alleges that each of the defendants infringes its U.S. Patent No. 4,992,432 ("the '432" patent") by "using, offering to sell, and/or by selling and/or importing into the United States application specific integrated circuits designed by or using information generated by, the process" described in the '432 patent.

Presently before the court is the defendants' motion to stay this action, or, in the alternative, to transfer this action to the United States District Court for the Northern District of California pursuant to 28 U.S.C. § 1404. For the following reasons, the court will grant the defendants' motion to transfer.

II. BACKGROUND

This case is a patent infringement action involving technology related to the design of

application-specific integrated circuits (“ASICS”). ASICS are microelectronic devices that electronics manufacturers design for a specific function, for example, for use in graphics and telecommunications devices.

The defendant Aeroflex Incorporated (“Aeroflex”) is a high technology company that designs, develops, manufactures, and markets a diverse range of microelectronics. Its principle place of business is in Plain View, New York. AMI Semiconductor, Inc. (“AMI”) is a high technology company that designs, develops, and manufactures a broad range of integrated circuit products for a number of end-uses. Its principle place of business is in Idaho. The Matrox defendants are high technology companies that design software and hardware solutions in the fields of graphics, video editing, image processing, and new business media. The Matrox defendants’ principle places of business are in Canada, New York, and Florida.

Ricoh is a high technology corporation that manufactures digital office equipment. Its principle place of business is in Japan. While the record does not reflect that Ricoh has any facilities in Delaware, it has six subsidiaries in California, including three within the Northern District of California.

Third-party Synopsys, Inc. (“Synopsys”) is a designer and manufacturer of high-level design automation solutions for the design of integrated circuits and electronic systems. Synopsys sells its products, including the Design Compiler at issue in the present case, to semiconductor, computer, communications, consumer electronics, and aerospace companies, including each of the defendants. Its principle place of business is in California.

On July 9, 2002, Synopsys filed a declaratory judgment action against Ricoh in the Northern District of California. Through that action, Synopsys seeks a declaration of non-infringement and

invalidity of the '432 patent.

III. DISCUSSION

Section 1404(a) provides that “[f]or convenience of [the] parties and witnesses, in the interest of justice,” the court may transfer a civil action “to any other district . . . where it might have been brought.” 28 U.S.C. § 1404(a). It is the movants’ burden to establish the need for transfer, and ‘the plaintiff’s choice of venue [will] not be lightly disturbed.’ *Jumara v. State Farm Ins. Co.*, 55 F.3d 873, 879 (3d Cir. 1995) (citations omitted).

When considering a motion to transfer, the court must determine ‘whether on balance the litigation would more conveniently proceed and the interest of justice be better served by transfer to a different forum.’ *Id.* This inquiry requires “a multi-factor balancing test” embracing not only the statutory criteria of convenience of the parties and the witnesses and the interests of justice, but all relevant factors, including certain private and public interests. *Id.* at 875, 879. These private interests include the plaintiff’s choice of forum; the defendants’ preference; whether the claim arose elsewhere; and the location of books and record, to the extent that they could not be produced in the alternative forum.¹ *Id.* at 879. Among the relevant public interests are: “[t]he enforceability of the judgment; practical considerations that could make the trial easy, expeditious, or inexpensive; the relative administrative difficulty in the two fora resulting from court congestion; the local interest in deciding local controversies at home; [and] the public policies of the fora.” *Id.* at 879-80 (citations omitted).

¹ The first three of these private interest collapse into other portions of the *Jumara* analysis. The court, therefore, will consider them in the context of the entire inquiry only. See *Affymetrix, Inc. v. Synteni, Inc. and Incite Pharmaceuticals, Inc.*, 28 F. Supp. 2d 192 (D. Del. 1998).

In the present case, Ricoh disputes that this action could have been brought in the Northern District of California due to a lack of personal jurisdiction over the defendants. While it is not this court's province to determine another court's jurisdiction, and it therefore expresses no opinion on this subject, the court will, however, note the following. The defendants assert that they have each registered to do business with the California Secretary of State or have solicited and made allegedly infringing sales in California. Additionally, as the defendants themselves have stated that litigating this action in California would be more convenient and preferable to them, it does not appear that a California court's exercise of personal jurisdiction over them would offend the minimum requirements inherent in the concept of fair play and substantial justice.

Ricoh next contends that it would be improper to transfer this action to California because the Delaware action is first-filed. While the court does not dispute that this action is first-filed, it concludes that an exception to this rule controls the present inquiry. Under Federal Circuit precedence, a manufacturer's declaratory judgment suit should be given preference over a patentee's suit against the manufacturer's customers when those customers are being sued for their ordinary use of the manufacturer's products. *See Katz v. Lear Siegler, Inc.*, 909 F.2d 1459, 1464 (Fed. Cir. 1990). This rule, known as the "customer suit exception," recognizes that it is more efficient for the dispute to be settled directly between the parties in interest. *See Whelan Tech., Inc. v. Mill Specialities, Inc.*, 741 F. Supp. 715, 716 (N.D. Ill. 1990) (noting that the manufacturer is presumed to have a greater interest in defending its actions against charges of patent infringement.) It also acknowledges that a patentee's election to sue customers, rather than the manufacturer itself, is often based on a desire to intimidate smaller businesses. *See Kahn v. General Motors, Corp.*, 889 F.2d 1078, 1081 (Fed. Cir. 1989).

In the present case, the court concludes that Ricoh's infringement claims against the defendants are fundamentally claims against the ordinary use of Synopsys' Design Compiler. Thus, the California court's determination regarding infringement and validity of the '432 patent will efficiently dispose of the infringement issues regarding Synopsys' customers in this case. It is clear that, based on the outcome of the California case, either Synopsys will prevail and use of the Design Compiler will be determined to be non-infringing, or Ricoh will prevail, and Synopsys will be forced to pay damages or license the patent. In the latter situation, Synopsys' customers would then be immunized from liability. *See Intel Corp. v. ULSI Sys. Tech., Inc.*, 995 F.2d 1566, 1568 (Fed. Cir. 1993). For these reasons, the court finds that the first-filed rule does not control the present inquiry.

Upon consideration of the remaining Section 1404 transfer factors, the court concludes that the balance of convenience tips heavily in favor of transfer. In reaching this conclusion, the court relied on the following considerations, among others: (1) no party maintains any facilities, personnel, or documents in Delaware; (2) no acts of alleged infringement have taken place in Delaware; (3) no relevant third-party witnesses, including Synopsys, reside in Delaware;² (4) both the present case and the case in the Northern District of California are in the relatively early stages of litigation; (5) the relevant industry, the electronic design automation industry, is located in the Northern District of California; and (6) any disparity in court congestion, to the extent there is any, is not so great as to weigh against transfer due to the action currently pending in the Northern District. Thus, the court finds that the public and private interests are sufficient to tip the balance

²Although Ricoh predicts that Synopsys will cooperate with whatever discovery of it is required in the present action, even though it is not a party, the court finds such an assertion suspect at best.

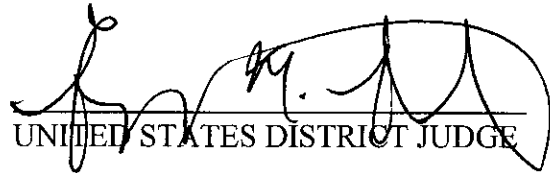
of convenience strongly in favor of transfer.³

IV. CONCLUSION

For the foregoing reasons, IT IS HEREBY ORDERED that:

1. The Defendants' Motion to Transfer (D.I. 67) is GRANTED.
2. Ricoh's Request for Leave to File a Sur-Reply (D.I. 122) is DENIED.
3. The above-captioned case is hereby TRANSFERRED to the United States District Court for the Northern District of California.

Dated: August 29, 2003



UNITED STATES DISTRICT JUDGE

³As a result of the court's decision to transfer this case, it will not decide the Section 271(g) discovery dispute presently pending before it.

1:03-cv-00103

Robert W. Whetzel, Esq. gp
Richards, Layton & Finger
One Rodney Square
P.O. Box 551
Wilmington, DE 19899
